Expert System for Specification Process in Software Development System Pure TELL

AUTHOR(S):

Saeki, Motoshi; Horai, Hisayuki; Enomoto, Hajime

CITATION:

Saeki, Motoshi ...[et al]. Expert System for Specification Process in Software Development System Pure TELL. 数理解析研究所講究録 1989, 709: 247-265

ISSUE DATE:

1989-12

URL:

http://hdl.handle.net/2433/101652

数理解析研究所講究録
第 709 巻 1989 年 247-265

Expert System for Specification Process in Software Development System Pure TELL

東工大·工·情工　佐伯 元司 (Motoshi Saeki)
富士通·国際研　蓬莱 尚幸 (Hisayuki Horai)
富士通·国際研　榎本 肇 (Hajime Enomoto)

ABSTRACT

The method to extract the structure of software modules based on object oriented model from a natural-language specification is presented. We have analyzed the process of constructing a formal specification based on an object-oriented model through several experiments, and collected the strategies for designing software modules into a knowledge base. We concentrate on types of verb patterns occurring in sentences and on types of relations among actions represented by verbs. In principle, each word such as nouns and verbs in a natural-language specification corresponds to a software module. Choosing candidate words for modules, determining kinds of them, and relating modules to other modules hierarchically depend on types of verb patterns and on types of the relations. These rules for extracting module structures are collected as some knowledge in our expert system for specification process, and we applies them to simple example "Floating Buoy System". For the target specification, we use the formal specification language of software development system called Pure TELL, which we are developing now. This language is based on strongly restricted English.

Keywords
Natural Language, Formal Specification, Software Process,
Expert system, Object Oriented Design

1. Introduction

To produce softwares of good quality, many researchers have studied methodologies of formal specification software design [5,7,11]. Almost of them, however, have not yet obtained as much success as we expected, because they gave only rough outlines and have left us to design the numerous mistakable parts of softwares. The process of specifying and designing softwares is one of the most intellectual and complex works done by human beings. Thus it remains to clarify when and what knowledge is used in the process, in contrast to the fields in which ordinary expert systems work well. Of course, making it clear will lead us not only to the improvement of software productivity but also to the the elucidation of the intellectual activity of human beings. Reusable software components [14] are considered concrete and simple domain-specific knowledge in the process, but many components are needed to produce practical softwares in the general area because reusable ones are too specific. We think that there are some domain-independent knowledge or generic strategies to specify and design softwares, and that they can be related to words occurring in informal requirement specifications of softwares written in natural language.

Although persons have only vague and ambiguous images of softwares which they

require, they can describe their images as informal specifications using natural language. Of course, the informal specifications may contain errors, and incomplete or inconsistent parts. One of the reasons why they can be written down in natural language is that the lexical and semantical structures of words in them are similar to the structures of the software components, i.e. modules. They are extracted from the substance of the images in natural form, and have many keys to their formal specifications or programs [6,7]. One of our aims is to establish a technique to obtain formal specifications from informal ones written in natural language. In this paper, we investigate what features of words, especially nouns and verbs, in natural-language specifications we take interest in to design the modules, and then extract a design strategy from the investigation results as knowledge. Furthermore we have implemented an expert system based on the strategy. By means of the interactions with users, which have understood informal requirement specifications, our expert system can acquire the information enough to construct their formal specifications. Our expert system generates queries based on the design strategy so that a suitable structure of software modules can be obtained.

In section 2, to extract a design strategy, we analyze and classify nouns and general verbs occurring in informal specifications. Especially we concentrate on verbs and their case structures, i.e. verb patterns [12]. Section 3 shows how to extract the structures of modules from informal specifications using features of words discussed in section 2. We use object oriented model to specify softwares formally. Section 4 gives an implementation of our expert system, and illustrates an formal specification of "Floating Buoy System" [5], to which our system is applied. It is written in Pure TELL specification language we are developing now. Its syntax is a strongly restricted and unambiguous English, and the specifications written in it are translated into temporal logical formulas automatically.

2. Verbs and Nouns in Informal Specifications

In this section, we analyze informal specifications written in English, and classify nouns and general verbs in them. Especially we concentrate on verbs and their case structures, i.e. verb patterns [12].

2.1. Relation between Words and Software Modules

As mentioned in section 1, a lump of natural concept which software contains is attached to each word in the sentences of its description written in natural language. We think that it is possible to decompose and refine hierarchically the complex concept of software into smaller concepts which the words have, by means of word-oriented formalizations [3]. More concretely, each word corresponds to a decomposed software modules, and the relations among the modules are determined by the occurrences of the corresponding words in the sentences. Formal specifications result after clarifying the meaning of all the words in the descriptions. Our hierarchical decomposition and refinement technique is based on lexical decomposition [1], that is to say, we explain the content of software in simple natural-language sentences and then, in the same manner, specify successively all the key words we used. Each key word defined in the refinement process is considered a decomposed software module.

What words do we pay attention to in specifications written in natural language?

Natural-language specifications are described as a collection of declarative sentences, and their meanings are determined by connection relations among their sentences and meanings of their sentences. Meaning of a sentence depends greatly on its main verb. A small set of verbs are used in specifications, and the patterns of sentences containing them are restricted. We call these patterns verb patterns. Thus it is useful to collect and classify verb patterns for exploring keys to formal specifications.

In addition to our lexical decomposition method, what model do we employ to view formally software systems in real world? In object oriented framework such as SMALLTALK-80 [13], a software system consists of several individual objects communicating with each other. Objects having some common features are grouped into a class and instances of a class are objects belonging to it. The specification of objects includes actions caused by receiving messages, so called "methods". Objects have peculiar attributes and their attribute values express their internal states. Their attribute values can be changed only by actions specified in them. We should consider the correspondence between words and software components in object oriented framework. Generally speaking, nouns and verbs correspond to objects or classes and to messages respectively [5]. Furthermore, as shown in Fig.1, it seems natural that in a typical sentence we let its subject and one of its objective words correspond to a sender and to only one receiver of the message denoted by its verb. But this correspondence does not necessarily derives structures of modules from natural-language specifications directly, because the specifications are not written based on object oriented discipline. So deeper analysis of nouns and verbs is needed. We use several examples of informal specifications in [5,10,11] for this analysis.

<subject> <verb> <objective word#0>
< *preposition*#1 > < *objectiveword*#1 >, ..., <preposition#n>
<objective word#n>

Users return a copy of book to the library [10].

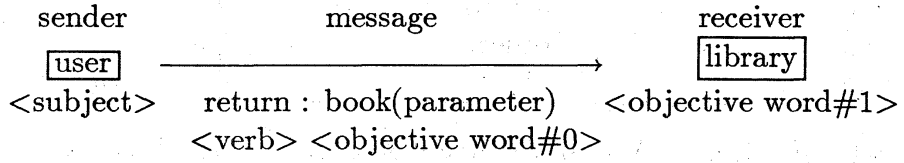


Fig.1 Correspondence between Sentence and Message sending

2.2. Classification of Nouns

We compose a sentence by filling slots of its verb pattern with nouns, noun phrase, or noun clauses. These nouns represent not only objects themselves but also other components in object oriented model. Classification of nouns is shown as follows.

1) Object noun : denoting an individual object
2) Class noun : denoting a class of individual objects
   A phrase consisting of an article and this noun, e.g. "the emergency switch" denote an individual object. These phrases are treated as object nouns.

Category 1) and 2) are the most popular use of nouns. Proper nouns and common nouns represent objects and classes respectively.

3) Attribute noun : denoting an attribute name or an attribute value of an object
The examples are common nouns "temperature" and "speed" in the phrases "temperature of the room" and "speed of the motor". These words are considered attribute names of the objects "room" and "motor" respectively. The values of "temperature" and "speed" express the internal states of their objects. In sentences, these nouns are often modified by nouns denoting an object.
4) Action noun : denoting an action of an object
Nouns derived from action verbs, e.g. "transmission" and "receipt" etc. belong to this category. In addition, we include gerunds and noun infinitives in the category. In the sentence "these(pressing buttons) cause the lift to visit the corresponding floor" [10], the noun infinitive "to visit" is the complement of "cause".

2.3. Classification of Verbs

We divide types of verbs into the following four categories based on their functions to nouns used together.

No.		Subject	Objective	Complement
I	1	object+	(value|object−)*	(value|object−)*
	2	value		
	3	object−	value$^+$	
	4	object−	object−, (value|object−)*	
	5	object−	object+, (value|object−)*	
	6	object−	object+, object+$^+$, (value|object−)*	
	7	object+	object+$^+$, (value|object−)*	
	8	object−	object+	value
II	1	object	action	
	2	object	object	action
	3	action		
	4	action	action	
	5	action	object	
	6	action	object	action

Table 1 Classification of Action Verbs and Action Relational Verbs

1) Relational Verbs
Relational verbs denote relations among objects. They need subjects and objectives, both of which represent objects. Relations among objects may change dynamically as executions of programs go, e.g. creation and deletion of objects. In the following example, the verb "have" is used as a relational verb and it denotes that a lift object is related to several button objects.
"Each lift has a set of buttons, one for each floor" [10]
2) Attribute Verbs
There are two types of attribute verbs, one denotes a declaration of an attribute name which a class of objects has, and need an objective word denoting the attribute name. Another type denotes attribute value of an object at a certain time. These verbs need complements denoting attribute values. Subjects in both type of verbs denotes objects.

3) Action Verbs
Action verbs denote actions, which changes attribute values of objects, i.e. internal states, or relations among objects. Of course, there are actions which do not cause to change internal states such as referring attribute values of objects. In usual sentences, agents of actions are their subjects and individual objects affected by the actions are their objective words or objective ones in their prepositional phrases which are necessary cases. Some sentences may contain several affected objects. There is the case where states of agents themselves are changed by their actions. We classify action verbs and action relational verbs, the fourth category of verbs, based on syntactic features and types of nouns used together. Table 1 shows the classification of them. We choose subject, objective, and complement as a syntactic feature. We include in objective category objective words in prepositional phrases affected by the actions. Object nouns are subdivided from a view of presence of their state-changes. "+" stands for nouns whose states are changed by actions, and "−" for no changes. "$\alpha \mid \beta$" stands for α or β. "α^+" stands for a repetition of α more than zero times, and "α^*" for a repetition of α more than or equal to zero times. For example, type I-8 verb has a subject, an objective, and a complement. The subject is an object noun whose state is not changed, the objective is an object noun whose state is changed, and the complement is a value of an attribute. There are two objects whose states are changed in the objectives of a type I-6 verb.

4) Action relational verb
Relations among actions (we call it action relation for short) such as temporal precedence relations are expressed by verbs in this category as well as adverbs. Especially verbs which have action nouns filled in their case slots often express relations. This type of verbs are shown in Table 1 by prefixing 'II'. These type II verbs denote actions or relations. In the sentence "he or she may flip a switch on the side of the buoy to initiate an SOS broadcast" [5], the verb "initiate" belongs to the type II-1 in Table 1. There are two interpretations of "initiate". One is the denotation of the action "initiate", which starts the action "broadcast an SOS message". The other is the causal relation between the action "flip a switch" and "broadcast", and in this case, there does not exist the action "initiate". Which of the interpretations is adopted depends on the context of the informal specification.

In the next section, we will present how to associate words with module components in object oriented model using this result of the classifications and action relations.

3. Structuring Modules based on Object Oriented Model

In this section, we explain how to extract the structures of modules from informal specifications using features of words discussed in section 2. We use object oriented model to specify softwares formally.

3.1. Action Verb Type and Actions

Classification of verbs are useful for extracting module components such as "classes" of objects, "attributes", and "methods". If we can consider that action verbs denote messages which cause the corresponding actions, a sentence containing an action verb as its main verb is a statement for sending a message. The problem is to decide

which of object nouns in the sentence stands for a sender or a receiver of the message. Since subjects are agents and objective words are individual objects affected by actions semantically, a subject and an objective word are considered a sender and a receiver respectively. The action verb rules in Table 2 are used to determine a sender and a receiver of the message corresponding to an action verb based on the above idea. For example, both the sender and the receiver of a message denoted by a type I-1 action verb are the subject of the sentence containing it. But there are two considerations in the correspondence as follows.

Action verb rule		
Verb type	Sender	Receiver
I-1	Subject	Subject
I-2	Object which has the attribute denoted by Subject	same as sender
I-3	Subject	Object which has the attribute denoted by Objective
I-4	case (a) there is an action relation about it : Subject | one of object− of Objective case (b) otherwise : delete it from the action category	
I-5	Subject	object+ of Objective
I-6	Subject	one of object+ of Objective
I-7	Subject	one of object+ of Objective or Subject
I-8	Subject	Objective
II-1	Subject	Objective of the action denoted by Subject
II-2	Subject	Objective
II-3	Agent of the action denoted by Subject	Objective of the action denoted by Subject

Action relation verb rule				
	Related action A		Another related action B	
Verb type	Sender	Receiver	Sender	Receiver
II-1	Subject	Sender of Objective	Sender of Objective	Receiver of Objective
II-2	Subject	Objective	Objective	Receiver of Complement
II-3	Sender of Subject	Receiver of Subject	Supplement another action	
II-4	Sender of Subject	Receiver of Subject	Sender of Objective	Receiver of Objective
II-5	Sender of Subject	Objective	Sender of Subject	Objective
II-6	Sender of Subject	Receiver of Subject	Objective	Receiver of Complement

Table 2 Rules for Verbs

1) Consider verbs which have more than one object nouns affected by the actions, e.g. I-3, I-6, I-7 and II-2 verbs in Table 1.
We call these verbs multiple verbs. Since message communications in object oriented model are one-to-one relations, a sentence should contain a sender and only one receiver, and the other words which are case components correspond to input or output data of the actions. Thus we should decompose a multiple

verb into several verbs, each of which contains only one noun affected by the corresponding action.
If the verb is a I-6 or I-7 verb, one of object+ (α) is the receiver. For each other object+ (β_i), we add the action whose sender is α and whose receiver is β_i. If the verb is a I-3 verb, we add as the same manner for each object which has the attribute denoted by objectives. For a II-2 verb, we add the action whose sender is objective and whose receiver is the objective of the action denoted by the complement.

2) Consider the directions of data flow in actions because the corresponding messages carry only input data of the actions.
In the sentence "The buoys collect air and water temperature, wind speed and location data through a variety of sensors" [5], it is considered that the sender of each "collect" message is "buoy" object and that the receiver is a "sensor". Other objective words such as "temperature" represent data carried by the messages "collect". These data, however, move from the corresponding "sensors" to "buoy". If we encounter these verbs, we should exchange the senders for the receivers, or add a new action which carries the data to the sender.

3.2. Action Relation

In order to specify an action relation between an action V1 and V2, a part of objects participating in an action V1, i.e. either the sender or the receiver of V1, should be common to a part of objects in an action V2. The definitions of actions should be encapsulated in the specifications of objects, i.e. "class" specifications. There are four patterns in relations among objects participating in two actions. These patterns are expressed schematically from case a) to case d) in Fig.2 (a). We should delete all undesirable patterns by exchanging the senders or receivers of the actions, or by replacing the corresponding action relations by relations among other actions, so that every action relation can be in either case a), b), or c) according to its type. What senders or receivers are exchanged and by what action relations are replaced also depend on types of action relations. We classify types of action relations into the following three categories by analyzing all kinds of words in the specification examples.

1) Causal relation : representing the temporal precedence relation between actions. This contains eventuality property of actions. In the sentence, "Speed readings are taken every 30 seconds" [5], read actions of speed values have a causal relation to themselves.
2) Exclusive relation : representing the prohibition of an action in a certain duration, e.g. mutual exclusion.
The sentence "minimum time for furnace restart after prior operation is 5 minutes" [10] stands for the prohibition of "start" action to the furnace during 5 minutes after the previous action.
3) Interruption relation : representing the priority of an action to another action, e.g. interruption, and exception.
The sentence "this (i.e. broadcasting information in response to requests from passing vessels) takes priority over the periodic broadcast" [5] is an example.

We have collected the rules for deleting undesirable patterns and they are called Action relation rules. Action relation rules are divided into three sub rules, Causal

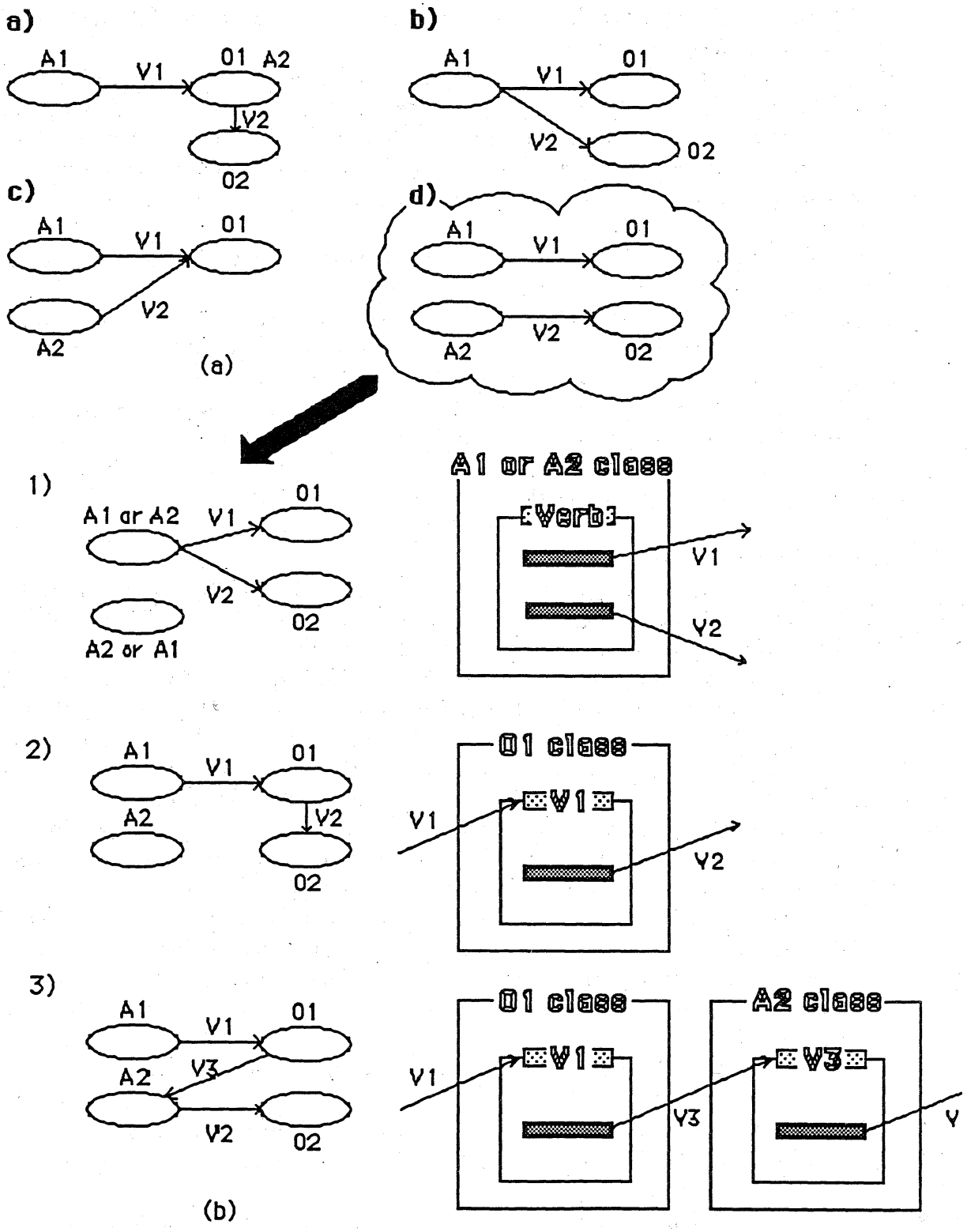


Fig.2 Causal Relations

relation rules, Exclusive relation rules, and Interruption relation rules. For example, consider the case of the causal relation that the action V1 precedes temporally the action V2. Case c) and d) in Fig.2 (a) are undesirable patterns because their action relations cannot be specified in ordinarily object-oriented framework. Precedence relations can be expressed only in the form of "procedure call" or "sequential execution" in the framework. If the action V2 preceded the action V1 in the case b), it would be an undesirable one, too. We should apply some rules called causal relation rules to the occurrences of these patterns. There are three rules illustrated in Fig.2 (b) :
[Causal relation rules]

1) Let both of the senders of the V1 and the V2 be the same, i.e. transform its occurrences into the pattern of case a) in Fig.2 (a). The action "Verb" which contains V1 and V2 defines the precedence relation between them using sequential execution form in procedural language.
2) Let the sender of the action V2, which follows the action V1, be the object O1, which is the receiver of V1. The "method" of the action V1 has the statement of sending the message corresponding to the action V2.
3) Add the new action V3, which precedes V2 and follow V1. The sender and the receiver of V3 are the receiver O1 of V1 and the sender A2 of V2 respectively. Note the two causal relations of V3 are specified in the form of 2) above.

We have described only causal relations briefly on account of limited space. In the cases of exclusive relation and interruption relation, we should equalize the receivers of the two actions to each other, i.e transform their occurrences into the pattern of case b) in Fig.2 (a).

3.3. Strategy

Our module structure extraction process are specified informally as follows.

1) Extract verbs from informal specifications and classify them. Nouns are also classified simultaneously into three categories in 2.1. Determine the semantical subjects and objective words of extracted action verbs, which are considered the candidates of senders and receivers respectively.
2) Extract action relations using keys of occurrences of type II-1$\sim$6 action verbs etc., and classify them into three categories in 3.2.
3) Search unnecessary nouns and verbs, e.g. synonyms, and delete them.
4) For each action verb or action noun, determine its sender and receiver using the following.
   i) Action verb rules
   ii) Action relation verb rules
   iii) Consideration 1) and 2) mentioned in 3.1.
5) For each undesirable action relation, apply the action relation rules to it and change the senders or the receivers. In the case of changing the sender of an action verb A, however, the following rules should be used in addition.
   [Sender Changing Rule]
   If only the former sender of A knows when the action A should occur, a new action verb B and a causal relation R should be created (shown in Fig.3). The sender and the receiver of B are the former sender and the new sender of A respectively. The relation R expresses that A follows B.

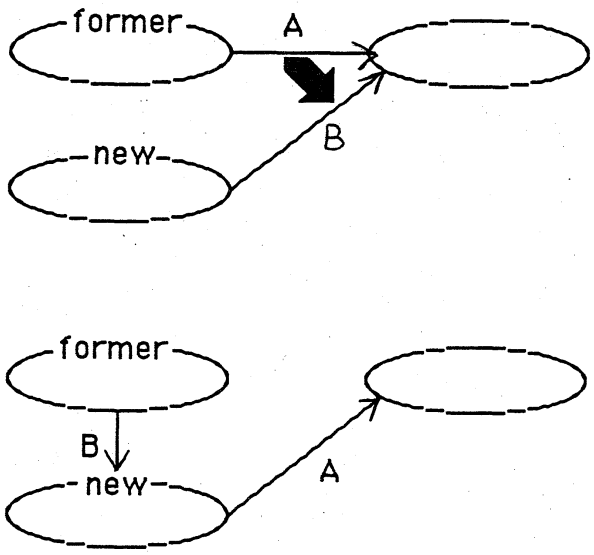


Fig.3 Sender Changing Rule

In step 4) and 5), there are several applicable rules, so users should select a rule to be applied out of them in the expert system. Of course, users do step 1)~3) as they are guided by the system. Individual objects and actions in software systems are determined according to the above rules.

4. Expert System for Structuring Modules

In this section, we explain an implementation of our expert system, and illustrate an formal specification of "Floating Buoy System" [5], to which our system is applied. It is written in Pure TELL specification language we are developing now. Its syntax is a strongly restricted and unambiguous English, and the specifications written in it are translated into temporal logical formulas automatically.

4.1. Formal Specification Language in Pure TELL [1,2,3]

Pure TELL system has been designed based on natural language (incl. lexical decomposition method) to support the whole of software development process. Using syntactic categories of words, Pure TELL captures the basic concepts of software, i.e. whether components in the software are 1) static or dynamic and 2) data or function/action as follows.

	static	dynamic
data, class	Common noun (Class) Proper noun (Value)	Common noun (Dynamic class) Proper noun (Object)
function, action	Noun, Adjective	General Verb (Action verb)

We use the word "dynamic" in the sense that timing of actions is essential in addition to input-output relations, e.g. concurrent systems and communication systems. In order

to specify dynamic systems, Pure TELL has the concept of "dynamic class", which is that of abstract data type with explicit timing specification, e.g. synchronization. Each action verb is defined as a "method" in a dynamic class.

Pure TELL specification language uses the notations of natural-language and figures. A sentence expression is a simple declarative sentence in the itemized form [1,2]. Each sentence is either a "static sentence" or an "action sentence". Every static sentence has be-verb as its main verb and represents a relation among objects denoted by its subject and objective words in its prepositional phrases. For example, "Delivering SOS-message is prior to delivering a weather data" in Fig.6 represents the relation between the instances of two kinds of "deliver" actions. Thus such relations correspond to adjectives, or common nouns which are not class names. Action sentences are used to specify dynamic systems, especially to define the bodies of "methods", and their main verbs are not be-verbs but general verbs representing actions. The action sentence "Radio receiver Rcv informs buoy B of request m" in Fig.6 denotes sending the message "inform" from the "radio receiver" object to the "buoy" object together with the parameter "m". Action sentences are connected based on the procedural control structures, i.e. "sequential execution", "conditional branch", and "repetition".

The semantics is also restricted so that each sentence can have only one and the most natural meaning. It is translated into a temporal logical formula by the method based on Montague Grammar[2,9].

4.2. Implementation of Expert System

We apply our module extraction technique to its own informal specification in 3.3 and derive the structure of the modules as a part of a formal specification of our expert system for specification construction. This specification is a part of the specification of specification process and written in Pure TELL language. We have translated semi-automatically the formal specification into the Prolog program, which is a prototype program of the system, through the temporal logical formulas. Thus we needed only tune up them and add the interface with users and other softwares, e.g. structured editor for the specification language. Consequently the knowledge, i.e. strategy to design softwares are declaratively represented in temporal logic, which gives strict meaning to Pure TELL descriptions. Temporal logical formulas are operationally interpreted by Pure TELL translator to Prolog, so that the system holds the strategy essentially in the form of production rules during the executions. This system is embedded in the structured editor, which have two kind of editors, text and graphics, and used from in the editor. The graphic editor has a faculty for semi-automatic arrangement of graphic components.

4.3. Free Floating Buoy Example

In this section, we illustrate an formal specification of "Floating Buoy System" [5]. This is a example to which our system is applied.

4.3.1. Extracting Objects and Actions

We will mention how to apply our module structure extraction technique to Free Floating Buoy System as an example. First we classify verbs occurring in an informal specification [5] of Buoy System while making up for omitted agents and/or individual

BUOYS SYSTEM

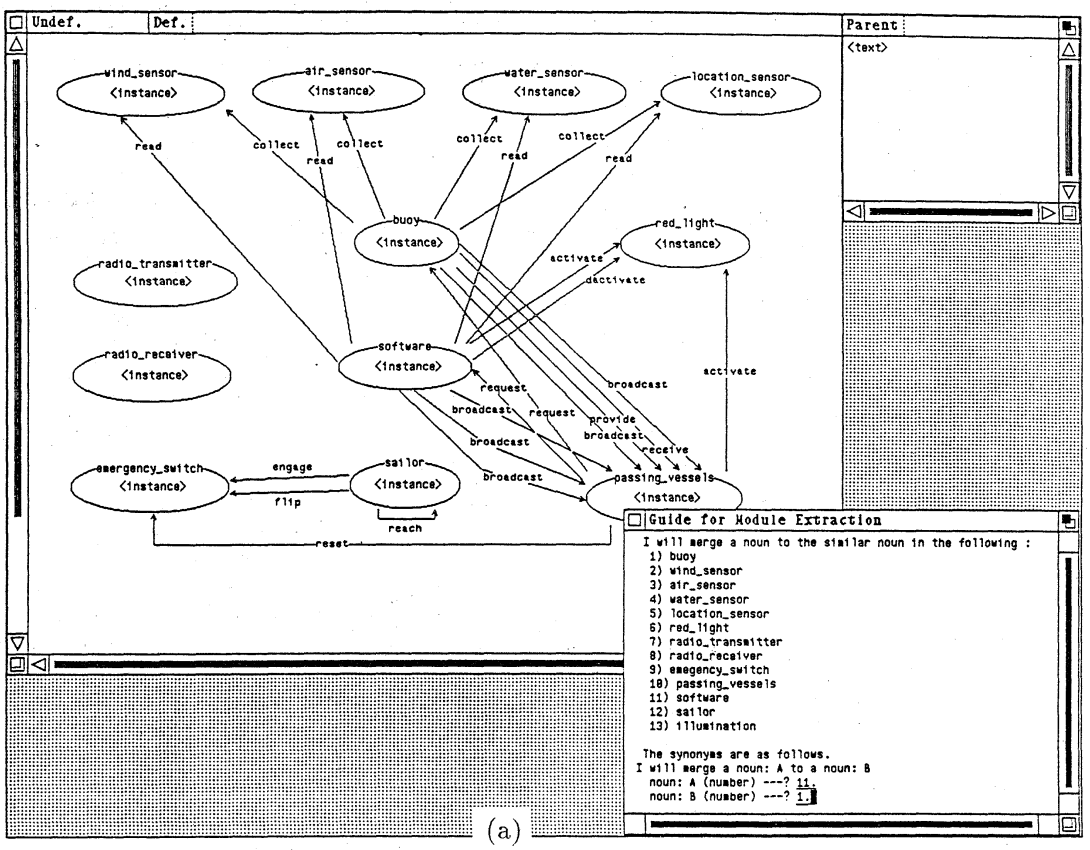
There exists *(1)* a collection of free-floating buoys that provide *(3.I-5 sub:buoy obj:air and ship traffic)* navigation and weather data to air and ship traffic at sea. The buoys collect *(3.I-4 sub:buoy obj:sensor)* air and water temperature, wind speed, and location data through a variety of sensors. Each buoy may have *(1)* a different number of wind and temperature sensors and may be modified to support other types of sensors in the future. Each buoy is also equipped *(1)* with a radio transmitter (to broadcast *(3.I-5 sub:buoy obj:passing vessels)* weather and location information as well as an SOS message) and a radio receiver (to receive *(3.II-1 sub:buoy obj:passing vessels* requests *(3.I-4 sub:passing vessels obj:buoy)* from passing vessels). Some buoys are equipped *(1)* with a red light, which may be activated *(3.I-5 sub:passing vessel obj:red light)* by a passing vessel during sea-search operations. If a sailor is able to reach *(3.I-1 sub:sailor)* the buoy, he or she may flip *(3.I-5 sub:sailor obj:switch)* a switch on the side of the buoy to initiate *(4.II-1 sub:sailor)* an SOS broadcast. Software for each buoy must :

1) maintain *(3.I-1 sub:software)* current wind, temperature, and location information; wind speed readings are taken *(4.II-3 sub:reading)* every 30 seconds, temperature readings every 10 seconds and location every 10 seconds; wind and temperature values are kept *(3.I-1 sub:software obj:sensor or buoy complement:running average)* as a running average.
2) broadcast *(3.I-4 sub:software obj:passing vessels)* current wind, temperature, and location information every 60 seconds.
3) broadcast *(3.I-4 sub:software obj:passing vessels)* wind, temperature, and location information from the past 24 hours in response to requests from passing vessels; this takes priority *(4.II-4 sub:broadcasting obj:broadcasting)* over the periodic broadcast.
4) activate *(3.I-5 sub:software obj:red light)* or deactivate *(3.I-5 sub:software obj:red light)* the red light based upon a request *(3.I-4 sub:passing vessels obj:software)* from a passing vessel.
5) continuously broadcast *(3.I-4 sub:software obj:passing vessels)* an SOS signal after a sailor engages *(3.I-5 sub:sailor obj:emergency switch)* the emergency switch; this signal take priority *(4.II-4 sub:broadcasting obj:broadcasting)* over all other broadcasts and continues *(4.II-3 sub: broadcasting)* until reset *(3.I-5 sub:passing vessel obj:emergency switch)* by a passing vessel.

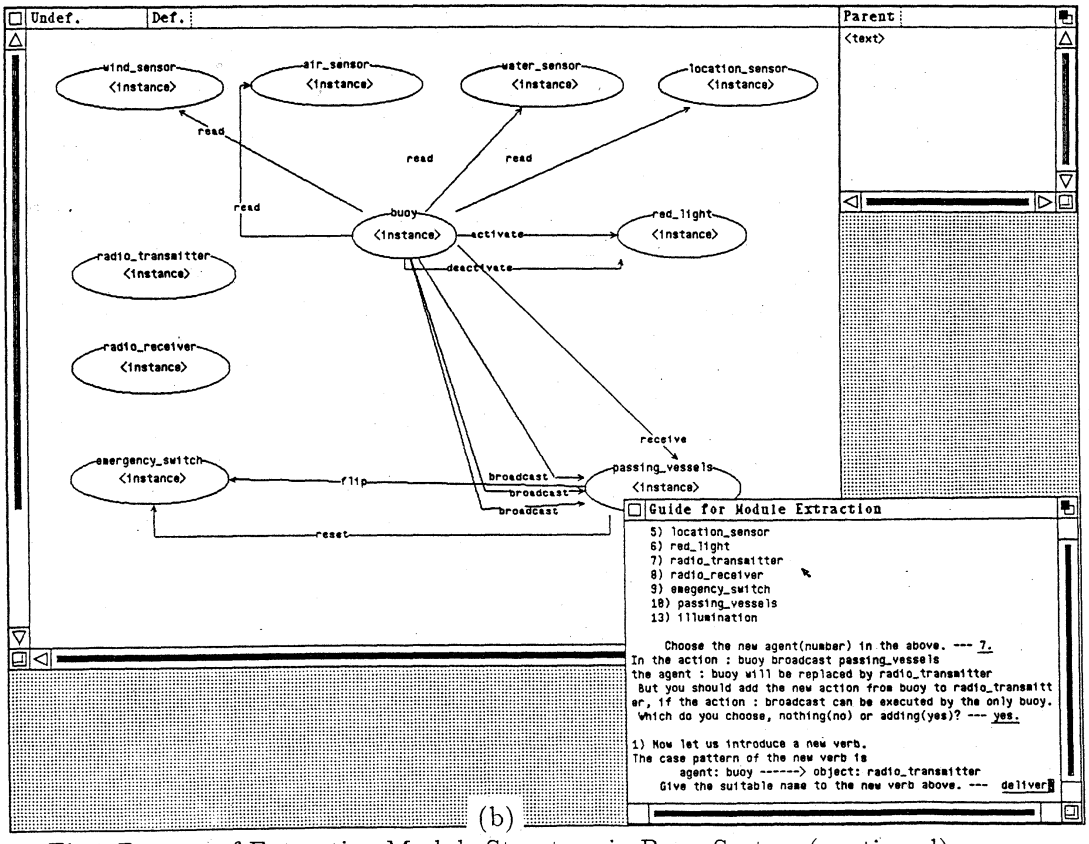
Action relations :

"collect"	precedes	"collect"
"read"	precedes	"read"
"request"	precedes	"receive"
"request"	precedes	"broadcast" (response to request)
"request"	precedes	"activate"
"request"	precedes	"deactivate"
"engage"	precedes	"broadcast" (SOS message)
"reset"	precedes	"broadcast" (SOS message)
"flip"	precedes	"broadcast" (SOS message)
"broadcast"	precedes	"broadcast" (periodic)
"broadcast"	precedes	"broadcast" (SOS message)
"broadcast" (SOS message)	interrupts	"broadcast" (response to request)
"broadcast" (response to request)	interrupts	"broadcast" (periodic)

Fig.4 Informal Specification of Buoy System and its Verb Classification

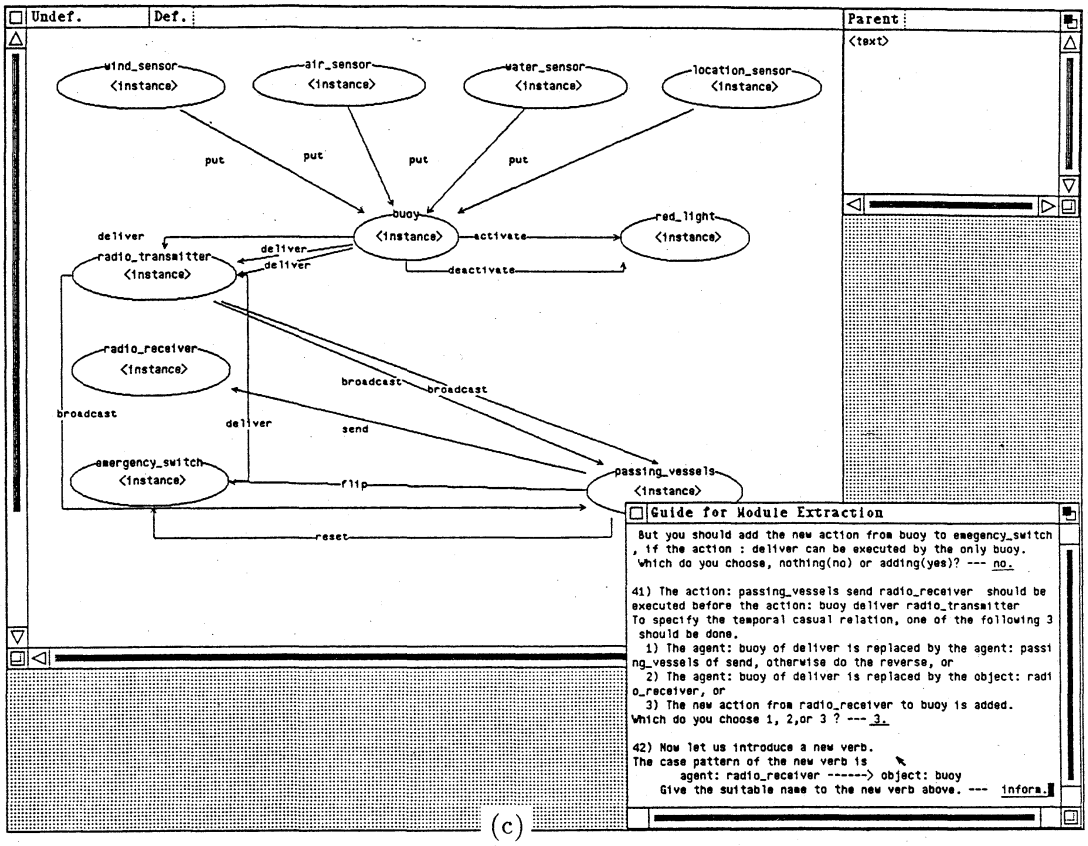


(a)

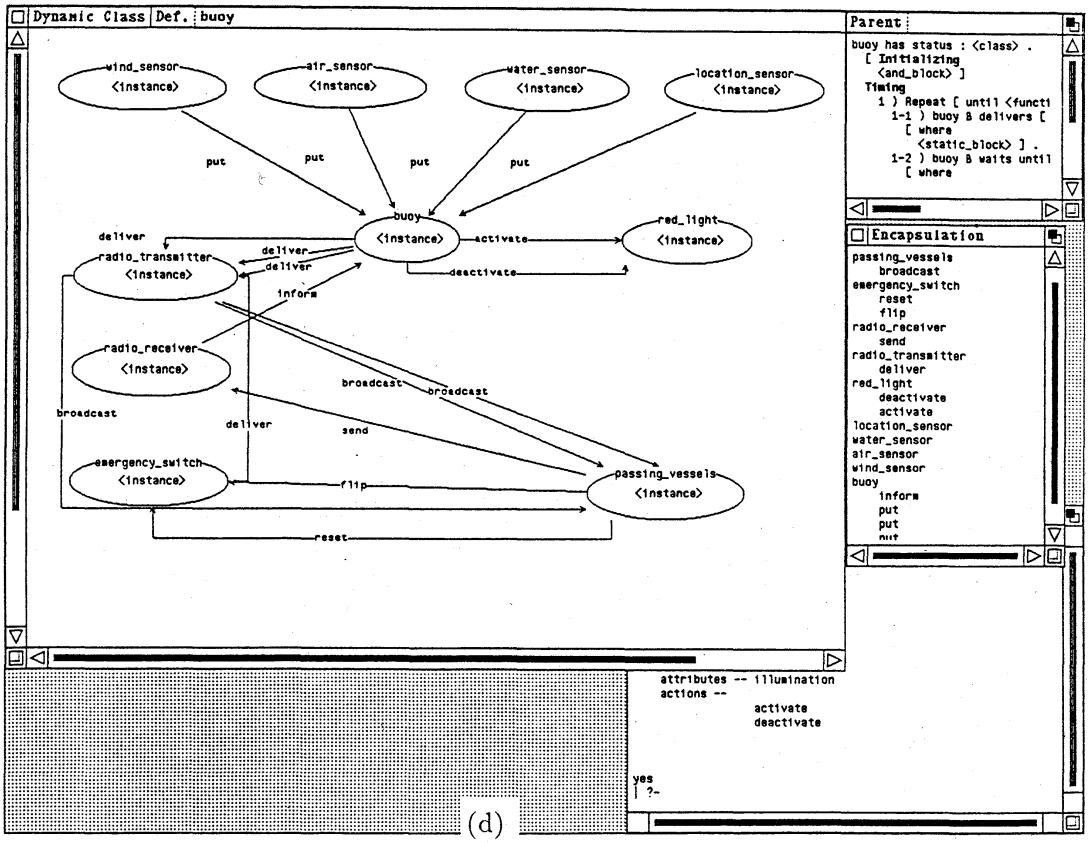


(b)

Fig.5 Process of Extracting Module Structure in Buoy System (continued)



(c)



(d)

Fig.5 Process of Extracting Module Structure in Buoy System

objects affected by the corresponding actions. We should transform passive form sentences into active form ones on account of extracting the agents and the objects of actions. Fig.4 shows the informal specification of Buoy System and the result of the classification of verbs. For example, "(3.I-4 sub:buoy obj:sensor)" means that the underlined verb "collect" belongs to the type I-4 in the category 3), i.e. action verb category, and that the semantical subject, i.e. agent and the affected object are "buoy" and "sensor" respectively. Fig.5 (a), (b), (c), and (d) shows the module structure extraction process using our system successively. In the graphical expression of the extracted objects and actions, each ellipse represents a candidate of an individual object denoted by a noun and each arrow stands for an action denoted by an action verb. The source and destination of an arrow are an agent and an object affected by the action respectively. We interpret verbs "initiate" and "continue" not as actions but as action relations.

To obtain the figure (b) from (a), we identify the unnecessary individual objects "software" and "sailor" with "buoy" and "passing vessels" respectively. The window of "Guide for Module Extraction" in Fig.5 (a) shows that the object "software" will be merged to "buoy". Underlined characters in the window stand for user-input ones. Periodic "broadcast" and "broadcast" requested from "passing vessels" are put together into the action "provide", so we delete "provide" as an unnecessary action. Similarly we delete the two "request"s and the "activate" from "passing vessels" to "red light".

Furthermore agents of three "broadcast"s and "receive" are changed to "radio transmitter" and "radio receiver" respectively. As shown in Fig.5 (b), each of these changes to "radio transmitter" causes to the addition of an action "deliver" according to Sender Changing Rule. In Fig.5 (b), the data flows of "read" and "receive" are opposed to the directions from their agents to their objective words, i.e. their action flows. Considering simplicity, we exchange their agents for their objective words. So, we change the action names "read" and "receive" to "put" and "send" respectively. The graphical part of Fig.5 (c) shows the result of the step above.

There is a causal relation between "send" and "deliver". To specify this relation, we add an action "inform" from "radio receiver" to "buoy" according to the causal relation rule 3). See the window in Fig.5 (c), in which rule 3) is selected. The addition of "deliver" from "emergency switch" to "radio transmitter" also results from the causal relation between "flip" and "broadcast". We finally have obtained Fig.5 (d). The "Encapsulation" window shows a structure of the extracted modules.

4.3.2. Formal Specification of Buoy System

A formal specification of Buoy System is shown in Fig.6, and we introduce the "weather data queue" object in order to hold data from the past 24 hours data. This object is in a lower layer in the hierarchical structure of object oriented model of Buoy System. Common nouns "buoy", "radio transmitter", "radio receiver", "emergency switch", various "sensor", "red light", "weather data queue" are defined as dynamic classes. "Passing vessels" is an environment of the system. The dynamic class "emergency switch" has the "flip" and the "reset" actions allowable from external objects. The meaning of the verbs are specified as "methods" in the dynamic classes. For example, "someone flips emergency switch E" declares the verb "flip", sender "someone", and receiver E belonging to the dynamic class "emergency switch". "Someone"

```
Buoy
  timing
    1) Repeat
      1-1) buoy B delivers periodically
             the current data of the weather data queue to the radio transmitter.
      1-2) buoy B waits for 60 seconds.

  Radio receiver Rcv informs buoy B of request m
    means that
      1) If request m is a past data request, then
           buoy B delivers weather data W to the radio transmitter.
        where 1-1) Weather data W is the data of the weather data queue according to
                   request m.
      2) If request m is a light activation request, then buoy B activates
           the red light.
      3) If request m is a light deactivation request, then buoy B deactivates
           the red light.
  end inform;

  Sensor S puts data x to buoy B
    means that
      1) Buoy B stores data x to the field F in the weather data queue.
      where 1-1) Field F is determined by sensor S.
  end put;

  Field F is determined by sensor S
    means that
      case 1) Sensor S is a wind speed sensor :  field F is wind speed.
      case 2) Sensor S is an air temperature sensor :  field F is air temperature.
      case 3) Sensor S is a water temperature sensor :
                field F is water temperature.
      case 4) Sensor S is a location sensor :  field F is location.
  end determined;

  Field is wind speed, air temperature, water temperature, or location.

  Weather data queue
    has wind speed :  queue of speed, air temperature :  queue of temperature,
        water temperature :  queue of temperature, location :  queue of location.

    Someone stores data x to field F in weather data queue Q
      means that
        1) The field F of weather data queue Q becomes data y.
        where 1-1) Data y is the result of shifting data x to data z.
              1-2) Data z is the value of the field F in weather data queue Q.
      end store;

      Weather data W is a current data of weather data queue Q
        means that
          1) Weather data W is speed S, temperature T1, temperature T2,
               and location L.
          where 1-1) Speed S is the last element of the wind speed of
                       weather data queue Q.
                1-2) Temperature T1 is the last element of the air temperature of
                       weather data queue Q.
                1-3) Temperature T2 is the last element of the water temperature of
                       weather data queue Q.
                1-4) Location L is the last element of the location of
                       weather data queue Q.
      end current data;
  end weather data queue;
end buoy;

Radio transmitter
  coordinating
    1) Delivering SOS-message is prior to delivering a weather data.
    2) Delivering a weather data is prior to delivering periodically.

  Someone delivers [periodically] information m to radio transmitter Tr
    means that
      1) Radio transmitter Tr broadcasts information m to the passing vessels.
  end deliver [periodically];
end radio transmitter;

Radio receiver
  Someone sends request m to radio receiver Rcv
    means that
      1) Radio receiver Rcv informs the buoy of request m.
  end send;
end radio receiver;
```

Fig.6 Formal Specification of Buoy System (continued)

```
Emergency switch
  has status :  on or off
  initializing
    1) The status is off.
  timing
    1) Infinitely
      1-1) If the status of the emergency switch is on, then
              the emergency switch delivers SOS-message to the radio transmitter.

  Someone flips emergency switch E
    means that
      1) The status of emergency switch E becomes on.
  end flip;

  Someone reset emergency switch E
    means that
      1) The status of emergency switch E becomes off.
  end reset;
end emergency switch;

Red light
  has illumination :  on or off
  initializing
    1) The illumination of the red light is off.

  Someone activate red light RL
    means that
      1) The illumination of red light RL becomes on.
  end activate;

  Someone deactivate red light RL
    means that
      1) The illumination of red light RL becomes off.
  end deactivate;
end red light;

Wind speed sensor is sensor of detection interval 30.
Air temperature sensor is sensor of detection interval 10.
Water temperature sensor is sensor of detection interval 10.
Location sensor is sensor of detection interval 10.

Sensor of detection interval i
  timing
    1) Infinitely
      1-1) The sensor puts data x to the buoy
              where data x is calculated.
      1-2) The sensor waits for i seconds.
end sensor;

Detection interval is positive integer.

Information is SOS-message or weather data.

Weather data is
  (wind speed ; speed, air temperature :  temperature,
      water temperature :  temperature, location :  location ).
```

```
[imported word]
  calculated in sensor
  passing vessel
    broadcast in radio transmitter
  SOS-message
  request
    past data request
    light activation request
    light deactivation request
  queue
    shift
```

Fig.6 Formal Specification of Buoy System

and "E" play a role of formal parameters. The successive part to "means that" is the module body and defines the meaning of the declared word by lexical decomposition. In the "inform" in the "buoy", this part consists of a set of action sentences, which are executed in order, i.e. "sequential execution". If an object has a value as an attribute, the "has" part of the dynamic class module specifies its domain. In the "emergency switch", the "status" is an attribute name and "on or off" is a domain specification, which expresses an enumeration of allowable values. A "timing" section in "emergency switch" specifies the sequence of actions which each "emergency switch" object causes itself. Thus the subjects of all the action sentences in the timing section should denote objects belonging to the dynamic class, i.e. this case is "emergency switch". A "coordinating" part in "radio transmitter" specifies the constraints among actions requested to the "radio transmitter" object, i.e. priority between two types of "deliver" actions.

5. Conclusion

We mentioned the strategy for extracting modules from natural-language specifications as knowledge. Our method should have more experience in specifying various kinds of softwares and be polished up together with classification of words. Another analysis of nouns are needed to support extraction of class hierarchy, e.g. super-sub class and meta class. Furthermore analysis and classification of words depend on what concepts are extracted from natural-language specifications. Fusing our approach to the technique of reusing specification components, i.e. domain-specific knowledge leads to constructing easily high-quality specifications. We are developing reusing technique based on natural-language words.

If a specification method has executable feature, specifying a software process allow tools supporting the process to be automatically generated [4]. Generated tools are used as prototypes to evaluate easily the software process. In fact, our system can be developed in a shorter period of time. But generated tools have human-interface poorer than practical tools. We should consider methodologies to include human-interface in a software process in future.

References

[1] Enomoto,H., Yonezaki,N., Saeki,M., Chiba,K., Takizuka,T. and Yokoi,T., *Natural Language Based Software Development System TELL*, Proc. of 6 th ECAI, pp.721-731 (1984)

[2] Enomoto,H., Yonezaki,N., Saeki,M. and Aramata,H., *Formal Specification and Verification for Concurrent Systems by TELL*, Proc. of 6 th ECAI, pp.732-745 (1984)

[3] Saeki,M., Horai,H., Toyama,K., Uematsu,N. and Enomoto,H., *Specification Framework Based on Natural Language*, Proc. of 4th International Workshop on Software Specification and Design, pp.87-94 (1987)

[4] Osterweil,L., *Software Processes Are Software Too*, Proc. of 9th ICSE, pp.2-13 (1987)

[5] Booch,G., *Object-Oriented Development*, IEEE Trans. on Soft. Eng., Vol.12, No.2, pp.211-pp.221

[6] Balzer,R., Goldman,N. and Wile,D., *Informality in Program Specification*, IEEE

Trans. Soft. Eng., Vol.4, No.2, pp.94-103 (1978)
[7] Abbott,R., *Program Design by Informal English Descriptions*, Commun. ACM, Vol.26, No.11, pp.882-894 (1983)
[8] Borgida,A. and Mylopoulos,J., *Knowledge Representation as the Basis for Requirements Specifications*, Computer, Vol.18, No.4, pp.82-91 (1985)
[9] Montague,R., *The Proper Treatment of Qualification in Ordinary English*, Reidel Dordrecht (1973)
[10]*Problem Set for the 4th International Workshop on Software Specification and Design*, Proc. of 4th International Workshop on Software Specification and Design, pp.ix-x (1987)
[11]Gehani,N. and McGettrick,A.D. (eds.) *Software Specification Techniques*, Addison Wesley (1986)
[12]Hornby,A.S., *Guide to Patterns and Usage in English, second edition*, Oxford University Press (1975)
[13]Goldberg,A. and Robson,R., it SMALLTALK-80 : The Language and Its Implementation, Addison Wesley (1983)
[14]*Special Issue on Software Reusability*, IEEE Trans. Soft. Eng., Vol.10, No.5 (1984)